

# Einführung ins Programmieren

Wintersemester 2024/25

Prof. Armin Scrinzi



*Hello World!*

*Terminal und Filesystem*

*std::cin, types, if-else*

*IDE*

*Loops*

*Container und auto*

*Die wichtigsten git-Befehle*

# Prüfungsrelevante Themen

Die folgenden Punkte sollten Sie auf jeden Fall für die Prüfung beherrschen

Die Liste wird im Lauf des Semesters ergänzt

Die endgültige Liste wird ca. 3 Wochen vor der Prüfung gepostet

- ✓ Kenntnis der Terminalbefehle
- ✓ Die wichtigsten git-Befehle
- ✓ Verwendung des Compilers g++
- ✓ Alles zur C++ Syntax
- ✓ Prinzipien des Programmierstils
- ✓ Prinzipien der Programmstruktur
- ✓ Alles zu Datentypen
- ✓ Alles zur Standard Library

## Was NICHT zur Prüfung kommt

- ✗ Verwendung von Visual Studio Code
- ✗ Gebrauch der Virtual Box

# **Organisation und Ablauf**

# Zweck der Vorlesung

...sicherzustellen, dass der gesamter Jahrgang der Physikstudierenden  
eine **Basis an Programmierkenntnissen** hat.

Programmieren ist ein Instrument der Physik in beinahe allen Bereichen,  
von vergleichbarer Bedeutung wie die **Mathematik**

Sie werden im Lauf Ihres Studiums programmieren müssen.

## Zum jeweiligen Thema in der VO:

- Vorstellung eines Beispiels
- Erklärung der Befehle und Strukturen
- Live-Demonstration und/oder Quiz
- Hausaufgaben: kleinere Programmieraufgaben

## Tutorien:

- Besprechen von Schwierigkeiten bei Ihrer Lösung der Hausaufgaben
- Hilfe bei Fertigstellung
- Diskussion einer “bestmöglichen” Lösung

# Inhalte (die ersten Schritte)

- "Hello world" - das elementarste Programm
- Das Command Line Interface (in Linux) – Verwendung des Terminals
- Schreiben, Compilieren und Ausführen eines Programms
- IDE - Integrated Development Environment
- Input und Output
- Variablen verschiedenen Typs: integers, floats, strings
- Wie finde ich Hilfe im Internet?
- class - Abbildung von Begriffen in die Programmiersprache
- C++ Syntax
- Adressen, referenzen, pointer, memory und ihre Beziehung zur Hardware
- Programm zur Plantenbewegung - numerische Integration einer Differentialgleichung
- Etc. etc...

Beginnen diese Woche, **ab morgen!**

**Aufgaben:**

werden jeweils nach der VO auf Moodle gepostet

**Lösungen:**

werden mit ca. 1 Woche Verzögerung gepostet

**Aber:** nur nützlich, wenn Sie's selbst probiert haben

**Termine:** siehe LSF

**Sprache:**

Mi, Do: Deutsch

Fr: **English!**

**Mittwoch ist überbucht! - Bitte umverteilen!**

Wenn nötig wird von uns umverteilt

## **Form:**

- schriftlich, ohne Zuhilfenahme elektronischer Mittel
- keine Handys oder sonstige Kommunikationsdevices

## **Art der Prüfungsaufgaben:**

- Wissensfragen
- Aufgaben ähnlich den Quizzes während der VO
- Aufgaben wie Tutoriumsaufgaben, soweit schriftlich sinnvoll

## **Termin:**

Bekanntgabe Anfang Dezember

# Umfrage Kenntnisstand

Programmiersprache	Keine: ~40%	Python,Java: ~ 50%
Terminal	Nein: ~ 65%	Ja: ~35%
Editor	Nein: ~50%	Ja: ~50%
ASCII/Binary	Nein: ~ 70%	Ja: ~30%

# Wie kann man diese Veranstaltung verwenden?

## **Keinerlei Vorkenntnisse**

Sie lernen Grundlegendes zum Umgang mit Computern,  
wie es in großen Teilen der Physik angewendet wird.

Sie lernen die Prinzipien einer Programmiersprache am Beispiel C++

## **Etwas Programmiererfahrung (z.B. in Python)**

Sie lernen eine der dominannten Sprachen (C++) kennen und  
typische allgemeine Eigenschaften des objektorientierten Programmierens  
sowie Regeln und Instrumente für effizientes Programmieren

## **Sie beherrschen C++ gut**

...Sie kriegen auf billige Art Ihre 3 ETCS ;-)

# Anmerkung in eigener Sache

Die Veranstaltung wird in dieser Form erstmals an der LMU angeboten

Es fehlen Erfahrungswerte, wie bestmögliche Ergebnisse zu erzielen sind

Wir versuchen in erster Linie ein Service für Sie zu liefern

Stellen Sie **Fragen**, machen Sie **Vorschläge** in der Vorlesung

Stellen Sie **Fragen**, machen Sie **Vorschläge** in den Tutorien

Nutzen Sie das **Studentenforum** in Moodle

Nutzen Sie das **anonyme Feedback** in Moodle

# Fragen

???

## Lektion 1

# Hello World

```
#include <iostream>
int main()
{
    std::cout << "Hello Student!" << std::endl;
    return 0;
};
```

- Virtuelle Linux Maschine
- Ein Programm im Editor schreiben
- Compilieren und Linken
- Das Programm ausführen

# Oracle Virtual Box



**Software-Simulation eines vollständigen Computers**

## BOX

Disc  
Memory  
Network  
CPU  
Keyboard  
  
etc. etc.

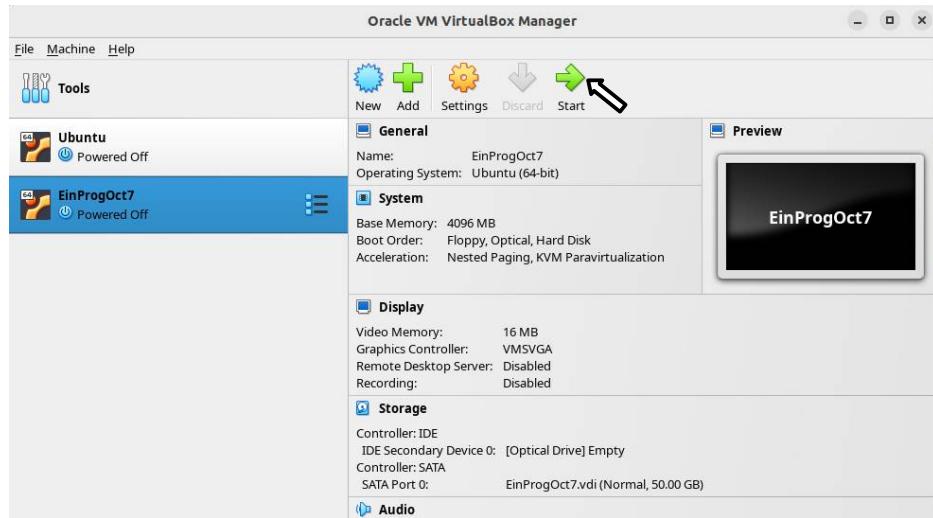
## Ihr Computer

- File
- Teil des Memory
- verbindet an die Netzverbindung Ihres Computers
- eine oder mehrere CPUs auf Ihrem Computer
- wird von Ihrem Computer durchgereicht

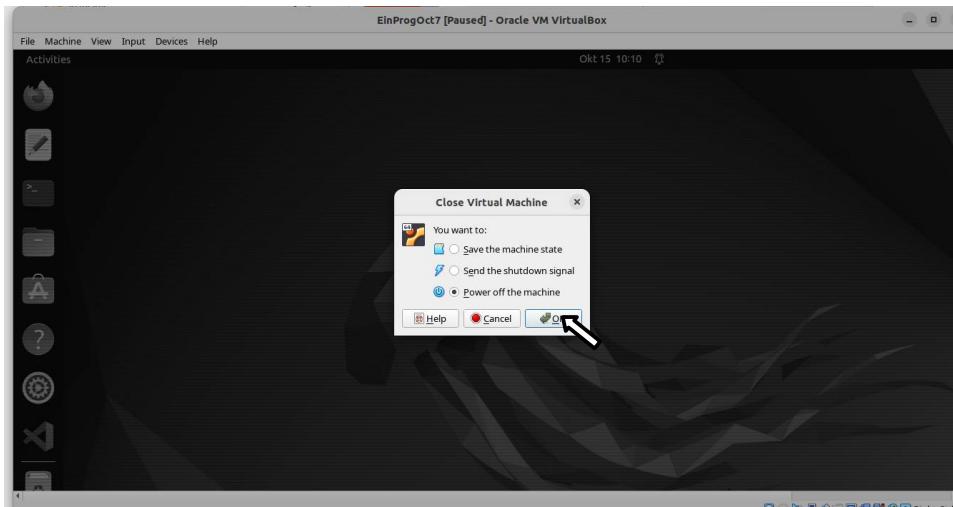
**Falls Sie aktiv an der VO teilnehmen wollen, müssen sie Virtual Box installieren**

(Unterstützung Online und in den Tutorien)

# Starten und Beenden der Virtual Ubuntu Box (VUB)



## Beenden: Fenster Schliessen



Wählen Sie “power off the machine”

# Das Programm

```
1 #include <iostream>
2
3 int main() {
4
5     std::cout << "Hello world!" << std::endl;
6
7     return 0;
8
9 }
```

Definiert input und output Operationen

Historisches Erbe: Programm ist die “main” Funktion

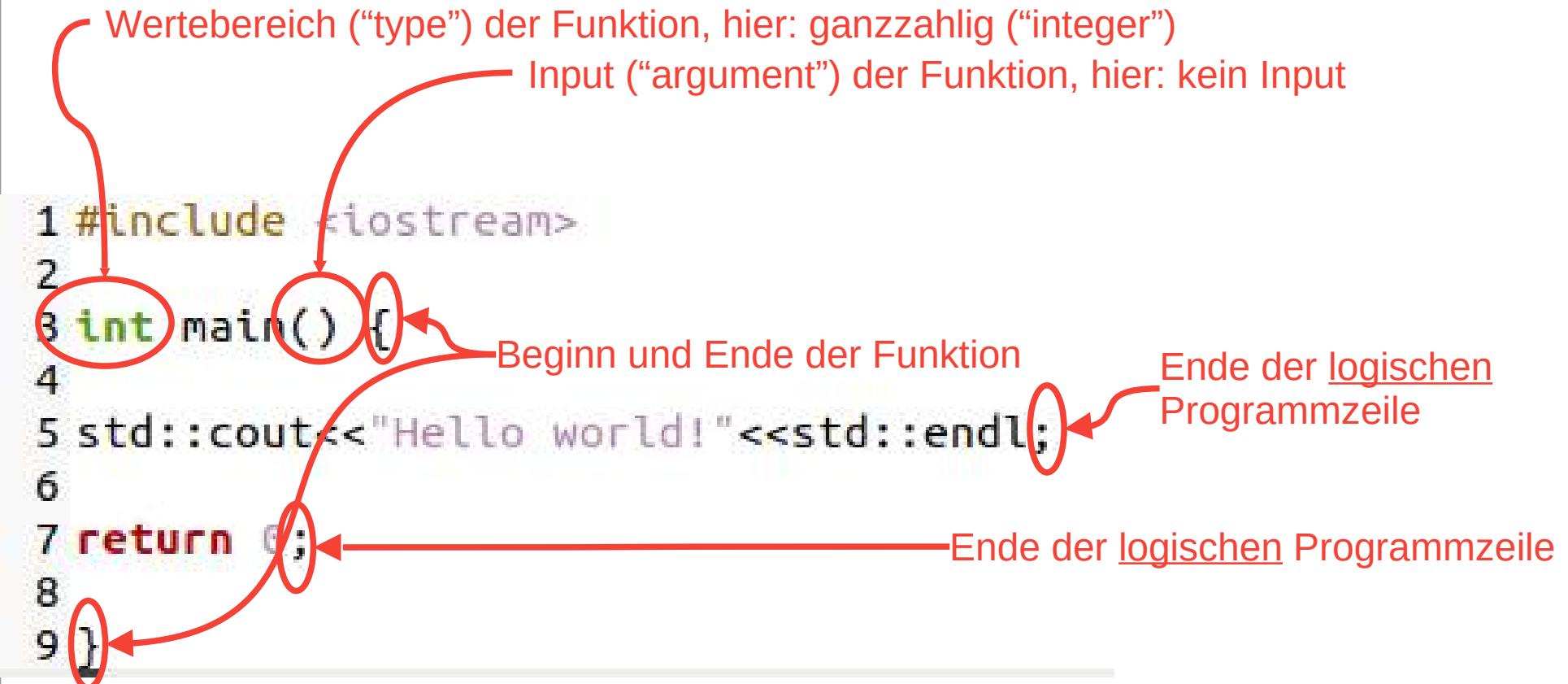
Wohin der Output erfolgt: an den “standard output”

“schiebt” end-line in den standard output

“schiebt” den Text in den standard output

Funktionswert 0 zeigt erfolgreiche Beendigung

# Wichtige Elemente der C++ Syntax



# Compilieren und Linken des Programms

Compilieren: Übersetzen des Programms in Instruktionen für die CPU

Linken: Verbinden der Instruktionen mit Hilfsprogrammen, z.B. für Input/Output

Bei einfachen Programmen meist beide Schritte gemeinsam

The screenshot shows a terminal window titled "student@EinProg: ~". The terminal displays the following command sequence:

```
student@EinProg:~$ g++ helloworld.cpp
student@EinProg:~$ ls
```

Annotations explain the steps:

- An arrow points from the word "g++" to the text "gnu c++ Compiler (default auf Linux maschinen)".
- An arrow points from the word "ls" to the text "Terminalbefehl: liste (ls) alle Files".
- A circled "a.out" file in the directory listing is annotated with the text "Executable – ausführbares Programm, erzeugt durch g++".

The terminal also shows a file named "helloworld.cpp" in the directory.

# Setze Programmnamen fest – “Compiler Flag”

```
student@EinProg:~$ Compiler flag
student@EinProg:~$ g++ helloWorld.cpp -o someStupidName
student@EinProg:~$ ls
a.out      Downloads      helloWorld.cpp    Loops
Class      FloatingPoint  IfElse           Multiply
Desktop   HarmonicOscillator IncorrectCode Screenshots
Documents HelloStudent    Libraries        snap
student@EinProg:~$ Selbstgewählter Name
someStupidName
```

## Allgemein: Hilfe zu Terminalbefehlen

```
student@EinProg:~$ student@EinProg:~$ g++ --help
Usage: g++ [options] file...
Options:
  -pass-exit-codes          Exit with highest error code from a phase.
  --help                     Display this information.
  --target-help              Display target specific command line options.
  --help={common|optimizers|params|target|warnings|[^\{][joined|separate|undocumented}\}][,...].
                           Display specific types of command line options.
  (Use '-v --help' to display command line options of sub-processes).
  --version                 Display compiler version information.
  -dumpspecs               Display all of the built-in spec strings.
```

# “Hello World!” in Python3

## Programm

```
1 print("Hello World!")
```

## Ausführen

```
student@EinProg:~$ python3 helloworld.py
Hello World!
student@EinProg:~$
```

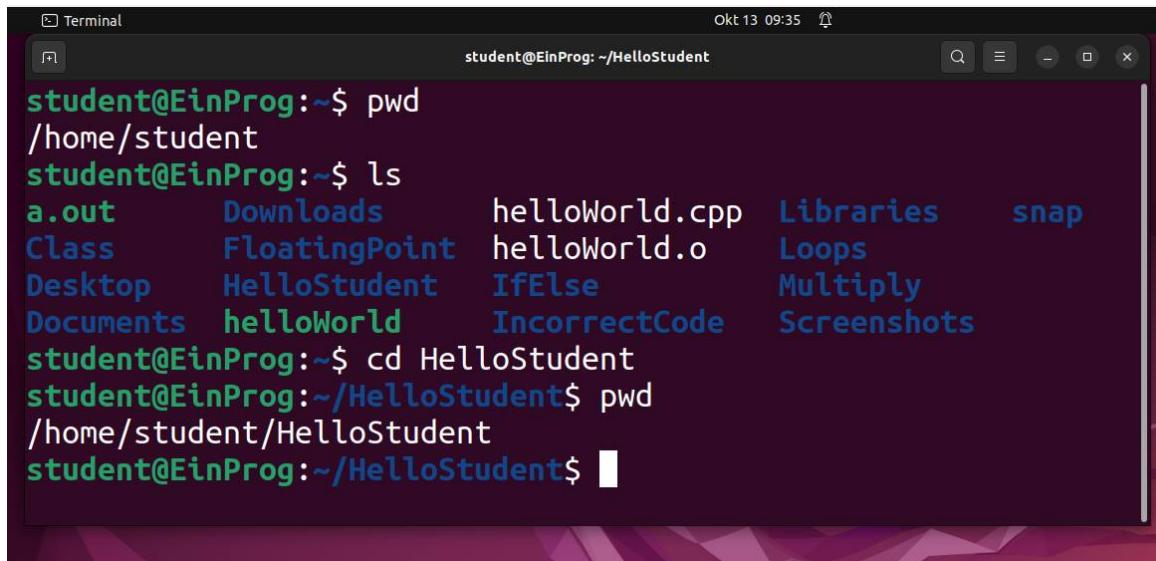
Keine Definitionen  
Kein Compilieren  
Eingängige Syntax

Viiiel netter....

**Problem: wenig Kontrolle, schlechte Performance**

## Lektion 2

# Terminal, File-System, Command line interface



A screenshot of a Linux terminal window titled "Terminal". The window shows a command-line session:

```
student@EinProg:~$ pwd
/home/student
student@EinProg:~$ ls
a.out      Downloads    helloWorld.cpp  Libraries   snap
Class      FloatingPoint helloWorld.o   Loops       Multiply
Desktop   HelloStudent  IfElse        Multiply
Documents  helloWorld   IncorrectCode Screenshots
student@EinProg:~$ cd HelloStudent
student@EinProg:~/HelloStudent$ pwd
/home/student/HelloStudent
student@EinProg:~/HelloStudent$
```

- bash (“Bourne again shell”) Command Line Interface (CLI)
- Bewegen im Terminal
- Files und Directories erzeugen und bearbeiten
- Wo finde ich Information und Hilfe?

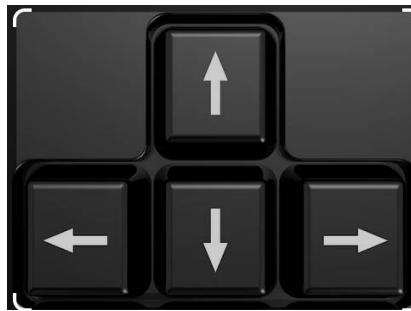
```
student@EinProg:~$ pwd ← Befehl: zeige "present working directory"  
/home/student ← Resultat: "home directory" für user "student"  
student@EinProg:~$ ls ← Befehl: liste Files und Directories im aktuellen Directory  
Desktop Documents Downloads HelloWorld Libraries snap  
student@EinProg:~$ cd HelloWorld ← wechsle ins Directory "HelloWorld"  
student@EinProg:~/HelloWorld$ ls ← ausführbares Programm (compiliert & linked)  
helloworld helloWorld.cpp ← C++ Code  
student@EinProg:~/HelloWorld$ ./helloworld ← Führe "helloworld" aus  
hello world ← Output von "helloworld" (nicht sehr spannend)  
student@EinProg:~/HelloWorld$ pwd  
/home/student>HelloWorld ← Wir befinden uns in diesem Directory  
student@EinProg:~/HelloWorld$ cd  
student@EinProg:~$ pwd  
/home/student ← Wir haben ins "home directory" gewechselt  
student@EinProg:~$ ls -l HelloWorld/helloworld.cpp  
-rw-rw-r-- 1 student student 95 Sep 12 20:18 HelloWorld/helloworld.cpp  
student@EinProg:~$ | lange ("-l") Info über helloworld.cpp im Directory HelloWorld
```

```
student@EinProg:~$ ls
Desktop Documents Downloads HelloWorld Libraries snap
student@EinProg:~$ mkdir dumDir ← erzeuge Directory "dumDir"
student@EinProg:~$ ls
Desktop Documents Downloads dumDir HelloWorld Libraries snap
student@EinProg:~$ cd dumDir ← wechsle nach "dumDir"
student@EinProg:~/dumDir$ mkdir dummerDir
student@EinProg:~/dumDir$ ls ← erzeuge dummerDir in dumDir
dummerDir
student@EinProg:~/dumDir$ cd dummerDir
student@EinProg:~/dumDir/dummerDir$ pwd
/home/student/dumDir/dummerDir Erzeuge leeres File "dumFile"
student@EinProg:~/dumDir/dummerDir$ ls
student@EinProg:~/dumDir/dummerDir$ touch dumFile ←
student@EinProg:~/dumDir/dummerDir$ ls ← wechsle in nächsthöhere Directory
dumFile
student@EinProg:~/dumDir/dummerDir$ cd .. ←
student@EinProg:~/dumDir$ ls ← wechsle in nächsthöhere Directory
dummerDir
student@EinProg:~/dumDir$ cd .. ←
student@EinProg:~$ ls entferne dumDir und alles darin
Desktop Documents Downloads dumDir HelloWorld Libraries snap
student@EinProg:~$ rm -r dumDir ←
student@EinProg:~$ ls
Desktop Documents Downloads HelloWorld Libraries snap
student@EinProg:~$
```

# Bewegen im Terminal

## Editieren einer Command Line

Links/rechts bewegen mittels Pfeilen  
Dann “backspace” zum Löschen



## Frühere Command Lines nochmal holen

Pfeil aufwärts – vorhergehende Command Line

Ctrl r *text* – erste vorhergehende Zeile die *text* enthält

Pfeil abwärts – wieder nach unten gehen

## Eingaben automatisch ergänzen

AnfangEinesBefehls TAB



# Zusammenfassung Command Line Interface (Terminal)

Online tutorials, z.B.

<https://ubuntu.com/tutorials/command-line-for-beginners>

## Tabelle aus Notizen (wird fallweise erweitert)

<code>ls</code>	“list”	zeige Files im aktuellen Directory
<code>pwd</code>	“present work dir.”	zeige den Namen des aktuellen Directory
<code>./prog</code>	führe im aktuellen Directory befindliches Programm <i>prog</i> aus	
<code>cd name</code>	“change directory”	wechsle in ins Subdirectory <i>name</i> ’
<code>cd ..</code>		wechsle ins nächsthöhere Directory
<code>vi</code>	“visual”	allgemeiner Linux Editor (auf jedem Linux)
<code>mkdir name</code>	“make directory”	Erzeuge neues Directory <i>name</i>
<code>rm name</code>	“remove”	File <i>name</i> entfernen
<code>rm -r name</code>	“remove recursive”	Directory <i>name</i> entfernen
<code>Ctrl c</code>	“control c”	bricht Programm ab
<code>Ctrl d</code>	“control d”	bricht Programm ab (alternative)
<code>Ctrl z</code>	“control z”	hält Programm an, ohne abzubrechen
<code>bg</code>	“background”	führt angehaltenes Programm im Hintergrund aus
<code>fg</code>	“foreground”	holt Programm in den Vordergrund
<code>Ctrl r text</code>	“recall”	suche vorhergehende Command Line, die <i>text</i> enthält
<code>partOfName TAB</code>		ergänze, oder zeige mögliche Ergänzungen für <i>partOfName</i>
<code>morefile</code>		zeigt Fileinhalt am Terminal (nur ASCII)

## Lektion 3

# Bausteine von C++: std::cin, Datentypen, if-else, Operatoren

```
#include <iostream>
int main()
{
    int a, b, prod;
    std::cout << "enter a: ";
    std::cin >> a;
    std::cout << "enter b: ";
    std::cin >> b;
    prod = a * b;
    std::cout << a << " * " << b << " = " << prod << std::endl;
}
```

- Code: multiply.cpp
- Der Datentyp int
- Dezimalzahlen (floating point): float, double
- Beispiele: multiply.cpp, ifElse.cpp
- Online Tutorials: <https://www.w3schools.com/cpp>

# multiply.cpp – 2 (ganze) Zahlen multiplizieren

```
#include <iostream>

int main()
{
    int a, b, prod;
    std::cout << "enter a: ";
    std::cin >> a;
    std::cout << "enter b: ";
    std::cin >> b;
    prod = a * b;
    std::cout << a << " * " << b << " = " << prod << std::endl;
}
```

Definition von a,b,prod als "int" – ganze Zahlen

Kein std::endl – folgender input in gleicher Zeile

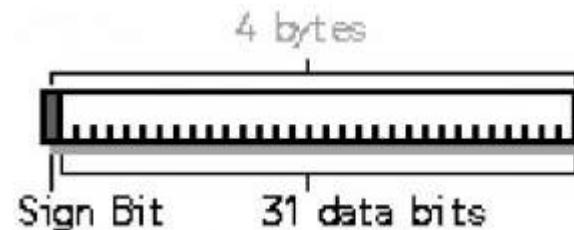
"standard C input stream" – normalerweise ans Terminal verbunden

"schiebt" Eingabe vom Terminal in Variable a

# Der Datentyp int

## Binäre Darstellung mittels 32 Bit = 4 Byte

1 Bit für das Vorzeichen + oder -  
31 Bit für den Zahlenwert



## Grösster darstellbarer Wert

$$2^{31} - 1 = \pm 2147483647$$

(Microquiz) Warum -1 ?

## Datentyp "unsigned int" für positive ganze Zahlen

Alle 32 Bit – grösster Wert 4 294 967 295

Anmerkung: **historisch** um Speicherplatz zu optimieren

Technisch: (leider) in der **standard library** häufig verwendet

Was geschieht wenn diese Werte überschritten werden?

# Ganzzahlige Datentypen in C++

Für die meisten Zwecke verwendet man int  
Ausser int gibt es noch zahlreiche andere ganzzahlige Datentypen  
C++ definiert dafür die **Minimalforderungen**

Type specifier	Equivalent type	Width in bits by data model				
		C++ standard	LP32	ILP32	LLP64	LP64
signed char	signed char	at least <b>8</b>	<b>8</b>	<b>8</b>	<b>8</b>	<b>8</b>
unsigned char	unsigned char					
short						
short int						
signed short						
signed short int						
unsigned short						
unsigned short int	unsigned short int					
int						
signed						
signed int						
unsigned						
unsigned int	unsigned int					
long						
long int						
signed long						
signed long int						
unsigned long						
unsigned long int	unsigned long int					
long long						
long long int						
signed long long						
signed long long int						
unsigned long long	unsigned long long int					
unsigned long long int	(C++11)					

Quelle:  
<https://en.cppreference.com/>

Verwenden sie die Datentypen gemäss dieser Definition

“Portabilität”

# #include <climits> Eigenschaften von integer types

Überprüfen der Werte am konkreten System

```
#include <iostream>
#include <climits>
int main(){
    std::cout<<INT_MAX<<"\tINT_MAX\tMaximum value for an object of type int"<<std::endl;
    std::cout<<INT_MIN<<"\tINT_MIN\tMinimum value for an object of type int"<<std::endl;
    std::cout<<CHAR_MAX<<"\tCHAR_MAX\tMaximum value for an object of type char"<<std::endl;
    std::cout<<"etc."<<std::endl;
}
```

Vollständige Liste auf cplusplus.com:

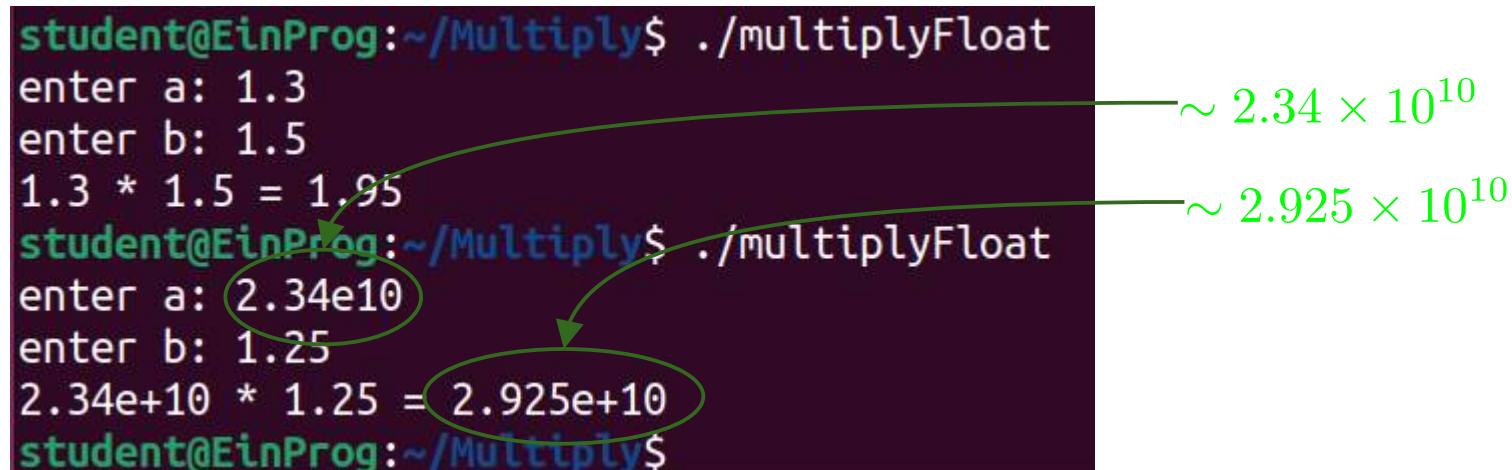
<https://cplusplus.com/reference/climits/>

# Dezimalzahlen – multiplyFloat.cpp



```
1 #include <iostream>
2 int main()
3 {
4     float a, b, prod;
5     std::cout << "enter a: ";
6     std::cin >> a;
7     std::cout << "enter b: ";
8     std::cin >> b;
9     prod = a * b;
10    std::cout << a << " * " << b << " = " << prod << std::endl;
11 }
```

Änderung zu multiply.cpp



```
student@EinProg:~/Multiply$ ./multiplyFloat
enter a: 1.3
enter b: 1.5
1.3 * 1.5 = 1.95
student@EinProg:~/Multiply$ ./multiplyFloat
enter a: 2.34e10
enter b: 1.25
2.34e+10 * 1.25 = 2.925e+10
student@EinProg:~/Multiply$
```

$\sim 2.34 \times 10^{10}$

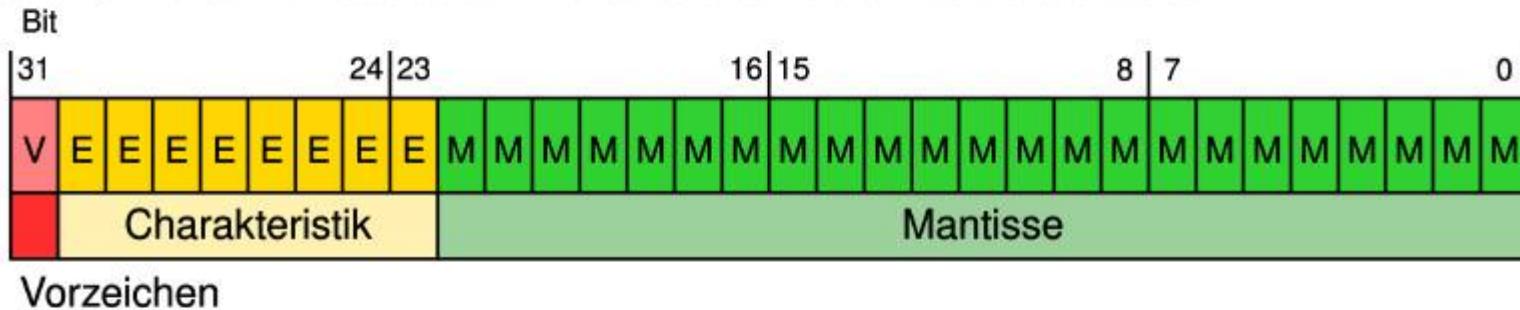
$\sim 2.925 \times 10^{10}$

# Dezimalzahlen – float und double

Eine Dezimalzahl mit endlich vielen Stellen kann als Paar zweier ganzer Zahlen (Exponent, Mantisse) dargestellt werden:

$$12.345 = \underbrace{12345}_{\text{Mantisse}} \times \underbrace{10^{-3}}_{\text{Basis}} \quad (1)$$

Dafür existieren Industriestandards, z.B. IEE 754 wo von 32 Bit 8 für den Exponenten und 23 Bit für Mantisse und 1 Bit für das Vorzeichen verwendet werden:



Damit ist die Maximalzahl der Dezimalstellen auf ca. 7 beschränkt und dezimale Exponenten bis  $\pm 38$  können dargestellt werden. Die 38 ergibt sich, weil nicht der dezimale, sondern der binäre Exponent verwendet wird  $2^{\pm 127} = 10^{\pm 127 \log 2} \approx 10^{38.2}$ .

Der IEE 754 für 64 Bit floating point Zahlen nutzt 11 Bit für den Exponenten, womit sich ca. 15 bis 16 Dezimalstellen und exponenten bis ca. 308 darstellen lassen.

Dies sind auch die Größenordnungen die sie für **float (32 Bit)** und **double (64 Bit)** auf üblicher Hardware erwarten können.

# if - else: Code mit Bedingungen steuern

Type "char" für "character": 1 Byte  $2^8=256$  Zeichen

```
char which;
std::cout << "enter case A or B: ";
std::cin >> which;
if (which == 'A')
{
    std::cout << "This is case A";
}
else if (which == 'B')
{
    std::cout << "This is case B";
}
else
{
    std::cout << "illegal case \\" << which << "\", allowed are: A,B";
}
std::cout << " [end of output line]" << std::endl;
```

Einfache Quotes, ein einzelner Buchstabe vergleiche "This is case A" - mehrer Zeichen

# bool - logische Variable und Operatoren, Vergleiche

Type "bool" für "boolean": 0 – false, 1 - true

```
int x = 5, y = 3;  
bool a, b, c, d;  
a = x < y;  
b = x == y + 2;  
c = y <= x - 2;  
d = y < x - 2;  
std::cout << "x < y: " << a << std::endl;  
std::cout << "x == y + 2: " << b << std::endl;  
std::cout << "y <= x - 2: " << c << std::endl;  
std::cout << "y < x - 2: " << d << std::endl;  
std::cout << " a and b: " << (a and b) << std::endl;  
std::cout << " a or b: " << (a or b) << std::endl;
```

Verknüpfung zwischen logischen Variablen

Siehe auch: [www.w3schools.com](http://www.w3schools.com)

[https://www.w3schools.com/cpp/cpp\\_operators\\_logical.asp](https://www.w3schools.com/cpp/cpp_operators_logical.asp)

# Operatoren: Algebra, Wertzuweisung

```
std::cout << a + b << std::endl;
std::cout << x + y << std::endl;
std::cout << a / b << std::endl;
std::cout << x / y << std::endl;
x = x + 2;
std::cout << x << std::endl;
a += 3;
std::cout << a << std::endl;
a++;
std::cout << a << std::endl;
x /= 3;
std::cout << x << std::endl;
```

Erhöht den Wert von a um 3

Erhöht den Wert von a um 1 ... C++

Dividiert den Wert von x durch 3

```
student@EinProg:~/Multiply$ g++ operators.cpp -o operators
student@EinProg:~/Multiply$ ./operators
```

```
21
21
4
4.25
19
20
21
6.33333
student@EinProg:~/Multiply$
```

# Operatoren: Algebra, Wertzuweisung

## +,-,\*,/ ... Algebra

### Division / für Typ int

```
int a=17, b=4;  
std::cout << a/b << std::endl;
```

Resultat: 4

Dezimalstellen werden dekappt ("truncated")

## = Zuweisung eines Wertes an eine Variable

Anmerkung:  $x = x + 2$  is eine korrekte Zuweisung (falsch als Mathematik)

## $+=, -=, *=, /=$ Veränderung des Variablenwerts

```
int a = 2;  
a += 3; // neuer Wert von a: 5  
a *= 4; // neuer Wert von a: 20  
// usw.
```

# Vermischtes (Fragen von nach der Vorlesung)

## Backslash / escape (string type)

Manche Buchstaben/Zeichen haben im String besondere Bedeutung  
z.B. die Anführungszeichen “ und ‘

Zum Schreiben im String:

```
std::cout<<" dies ist ein \"test\" \";
```

Output:

```
 dies ist ein "test"
```

Längere Liste von “escape sequences”:

<https://en.cppreference.com/w/c/language/escape>

## Programm abbrechen

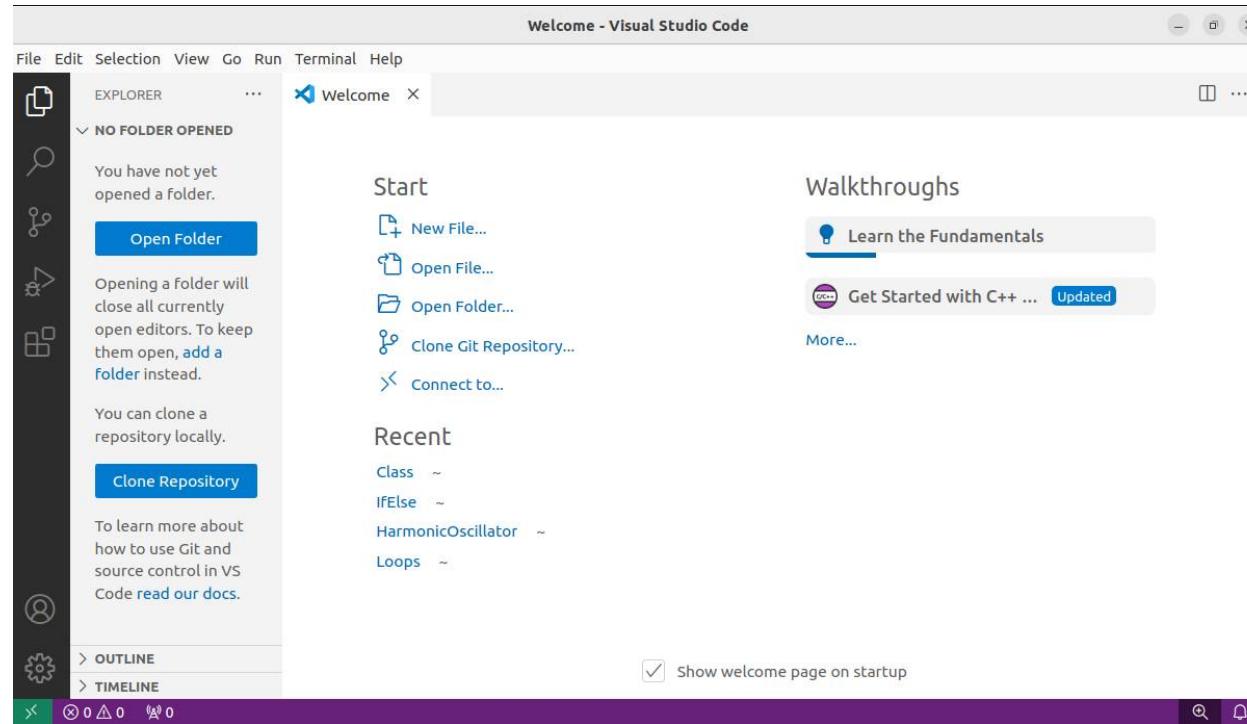
ctl-c      bringt fast alles um

ctl-d      auch wert zu versuchen, beendet oft hängende Eingabe

## Andere Fragen?

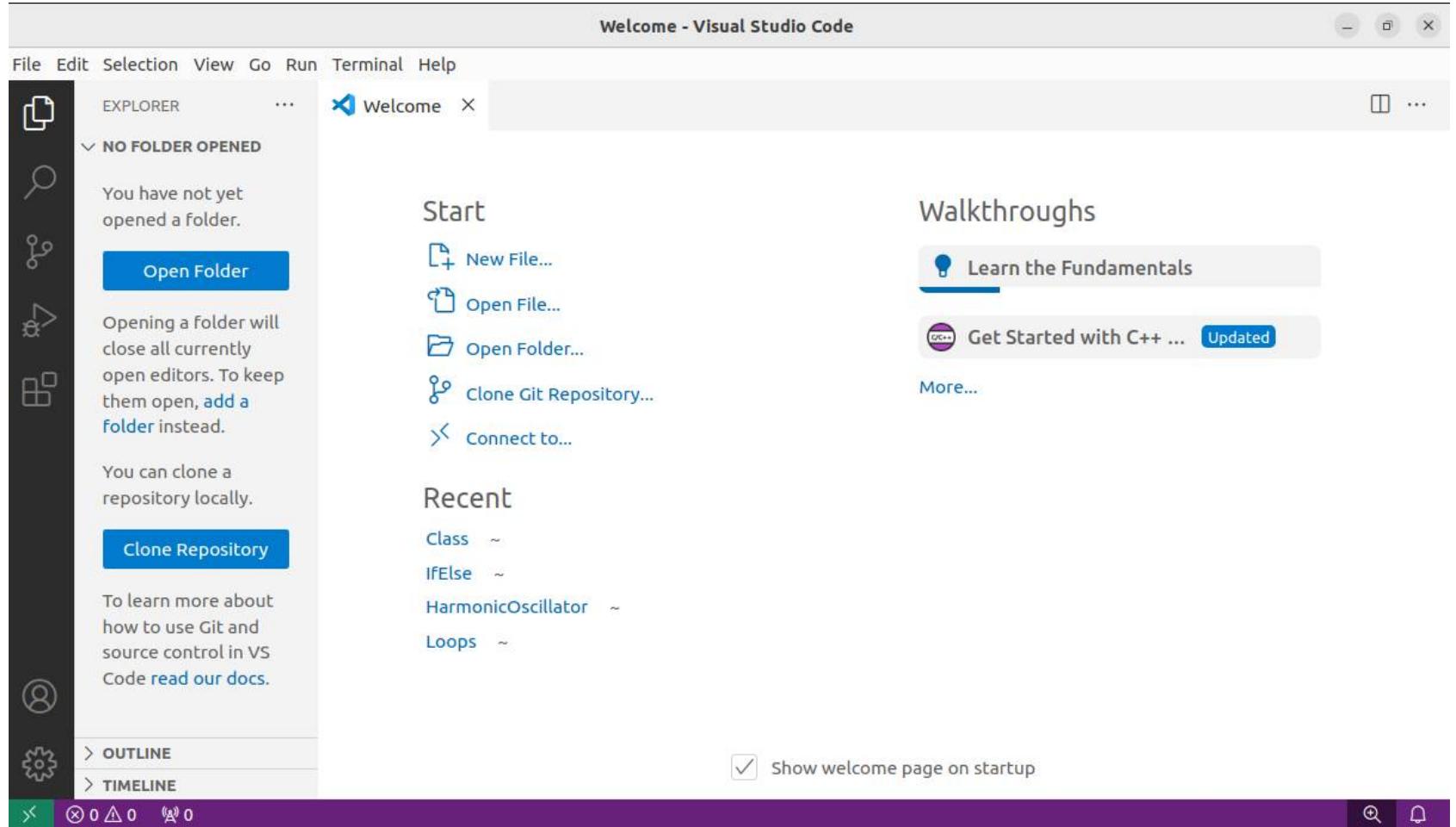
## Lektion 3

# IDE – Integrated Development Environment



- Visual Studio Code - starten und verwenden
- Syntaxchecks
- Standardformatierung von C++ Code

# Das Visual Studio Code Fenster

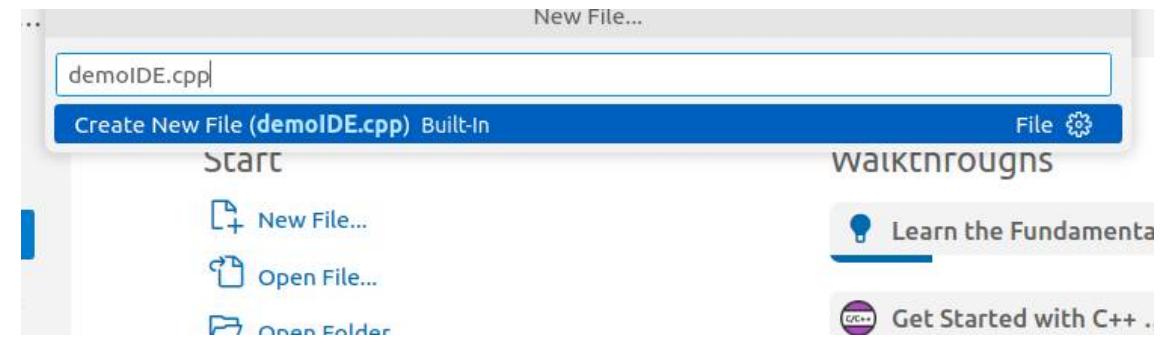


# Neues File

Neues File anlegen

- Start
- New File...
- Open File...
- Open Folder...
- Clone Git Repository...
- Connect to...

Filenamen eingeben



Directory auswählen oder neu erzeugen

student

- Name
- Codes
- Desktop
- Documents
- Downloads

Folder Name

IDE

Create

File im gewählten Directory anlegen

demoIDE.cpp

Search icon

Create File



Size

Type

Modified

# Code schreiben und compilieren

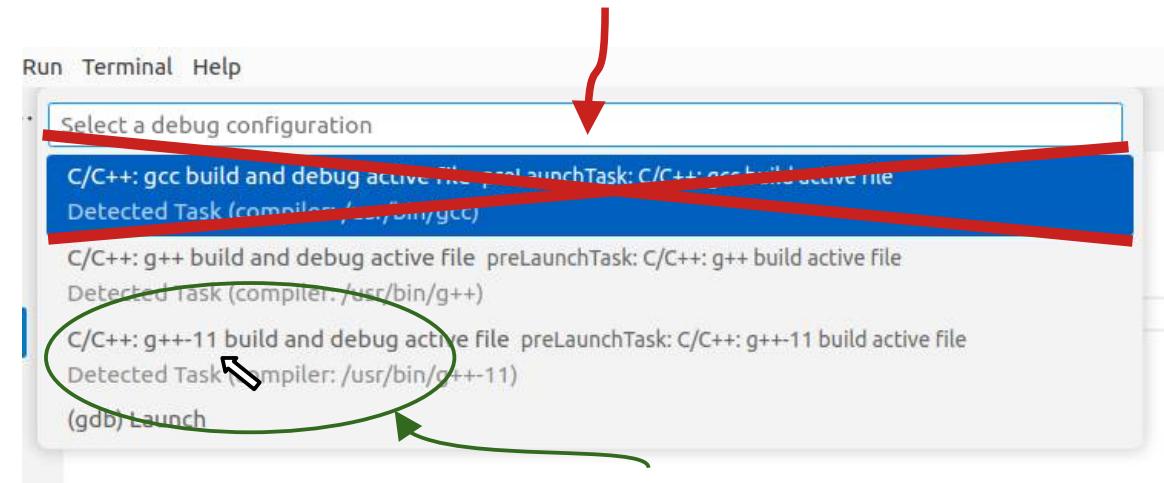
Editor mit erster Zeile



Compilieren



Compiler wählen



## Meldungen zum Compiliervorgang in DEBUG CONSOLE

```
demoIDE.cpp x
home > student > IDE > demoIDE.cpp > main()
1 #include <iostream>
2 int main () {
3     std::cout<<"hello"<<std::endl;
4 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, lex...) C/C++: g++-11 x
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Breakpoint 1, main () at /home/student/IDE/demoIDE.cpp:3
3     std::cout<<"hello"<<std::endl;
Loaded '/lib/x86_64-linux-gnu/libstdc++.so.6'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libc.so.6'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libm.so'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libgcc_s.so.1'. Symbols loaded.
[Inferior 1 (process 5817) exited normally]
The program '/home/student/IDE/demoIDE' has exited with code 0 (0x00000000).
```

The screenshot shows the Visual Studio Code (VS Code) interface. At the top, there's a navigation bar with 'File', 'Terminal', and 'Help'. Below it is a code editor window titled 'demoIDE.cpp' showing a simple C++ program:

```
home > student > IDE > demoIDE.cpp > main()
1 #include <iostream>
2 int main () {
3     std::cout<<"hello"<<std::endl;
4 }
```

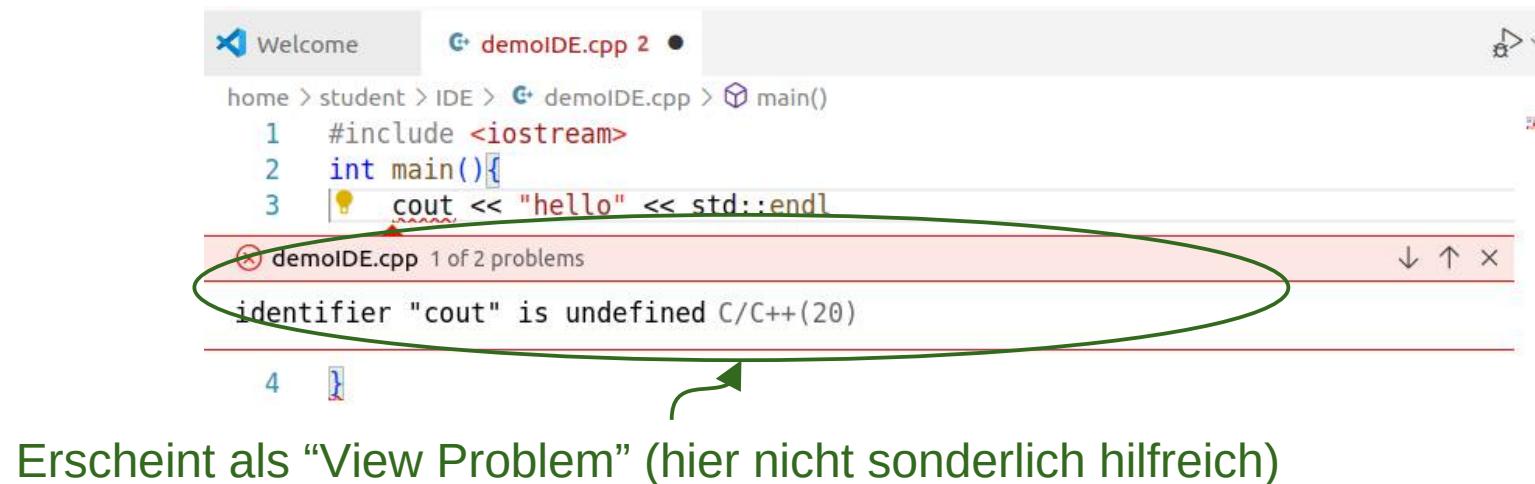
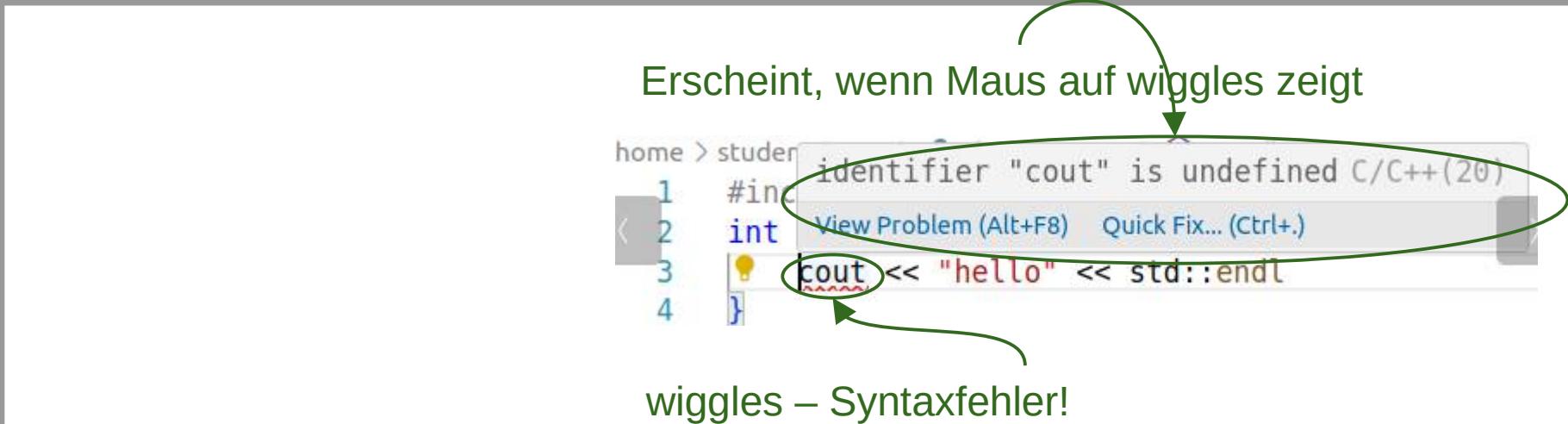
At the bottom of the interface, there's a tab bar with 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is highlighted with a green oval and arrow), and 'PORTS'. Below the tabs, a terminal window displays a session:

- student@EinProg:~\$
- student@EinProg:~\$
- student@EinProg:~\$ cd IDE
- student@EinProg:~/IDE\$ ./demoIDE
- hello
- student@EinProg:~/IDE\$

On the right side, there's a sidebar with a 'cppdbg: de...' entry and a 'g++-11 ...' entry with a checkmark. The status bar at the bottom shows 'Ln 3, Col 35' and other settings.

Terminal auswählen, normal verwenden

# Hinweise zu Syntaxfehlern



# Automatische Standardformatierung von Code

## Standardformatierung ist dringend empfohlen

- Lesbarkeit
- Maschinelle Vergleichbarkeit von Code:  
eliminiert triviale Änderung wie extra Zeilenumbrüche etc.

### Schlecht formatierter Code

uneinheitliche Zeileneintrückung

if-Block nicht optisch hervorgehoben

mehrere logische Zeilen nebeneinander

```
home > student > IDE > demoIDE.cpp > main()
1 #include <iostream>
2 int main () {
3     std::cout<<"hello"<<std::endl;
4     // badly formated code
5     if(1==0){ int a =2; double b=0; }
6 }
```

### Automatische Formatierung

ctl-a ...gesamten Code auswählen

ctl-k ctl-f ...formatieren

gleiche Einrückung bei gleicher  
logischer Ebene

if-Block klar lesbar

```
home > student > IDE > demoIDE.cpp > main()
1 #include <iostream>
2 int main()
3 {
4     std::cout << "hello" << std::endl;
5     // badly formated code
6     if (1 == 0)
7     {
8         int a = 2;
9         double b = 0;
10    }
11 }
```

separate Zeile für  
jede logische Zeile

# Kurzbefehle in Visual Studio Code

Ctrl-w	Editor schliessen
Ctrl-q	Visual Studio beenden
Ctrl-a	Gesamten Text auswählen
Ctrl-z	Edits rückgängig machen
Ctrl-k Ctrl-f	Ausgewählten Text formatieren

Alle Befehle sind auch im Drop-down Menue verfügbar!

# Pause - meine Fragen an Sie zum Stoff bisher...

Wieviele Byte hat der Datentyp int (zumeist)?

Was ist dann der grösste Wert?

Wie werden float gespeichert?

Was ist die Syntax von if – else?

## Lektion 4

# Loop – wiederholtes Ausführen eines Programmabschnitts

```
#include <iostream>
int main()
{
    for (int k = 0; k < 10; k++)
    {
        int square = k * k;
        std::cout << square << std::endl;
    }
}
```

- `for(...Bedingung & Werte...){ ...code Zeilen... }`
- Scope – Geltungsbereich von Variablen
- `do { ... code Zeilen ...} while (...Bedingung...)`
- Beispiele: `forLoop.cpp`, `harmonicOscillator.cpp`
- Physik: Hamilton'sche Bewegungsgleichungen

# for – Loop (Standardform)

## Initialisierung

Erzeugt Variable k mit Wert 0

```
for (int k = 0; k < 10; k++)  
{  
    int square = k * k;  
    std::cout << square << std::endl;  
}
```

## Ausführungsbedingung

Zeilen innerhalb {...} werden nur ausgeführt, wenn  $k < 10 \Rightarrow \text{true}$

## Nach Ausführung

wird dieser Befehl ausgeführt und Bedingung überprüft

Code wird wiederholt

Logisch äquivalenter Code  
(aber **sehr** schlechter Stil!)

```
// BAD CODE - for demonstration purposes only  
{  
    int k = 0;  
    BeginLoop:  
    if (k < 10)  
    {  
        int square = k * k;  
        std::cout << square << std::endl;  
        k++;  
        // use of goto is STRONGLY discouraged  
        goto BeginLoop;  
    }  
}
```

# scope { ... } - Geltungsbereich von Variablen

Variable die innerhalb von { ... } definiert werden, haben nur dort Bedeutung

Ausserhalb sind sie “undefined” und können nicht verwendet werden

- Verhindert irrtümliche Wiederverwendung von Variablen
- Gibt Speicherplatz frei, wenn nicht mehr benötigt

Eine Variable “goes out of scope”

k und square sind “out of scope”

Visual Studio Code  
zeigt Fehler

```
11 // BAD CODE - for demonstration puroposed only
12
13 int k = 0;
14 BeginLoop:
15     if (k < 10)
16     {
17         int square = k * k;
18         std::cout << square << std::endl;
19         k++;
20         // use of goto is STRONGLY discouraged
21         goto BeginLoop;
22     }
23
24
25
26 // scope: k and square are is not defined here
27 std::cout << k << square; // this is a syntax error
```

forLoop.cpp 2 of 2 problems

identifier "square" is undefined C/C++(20)

# do {...} while (...) Loop

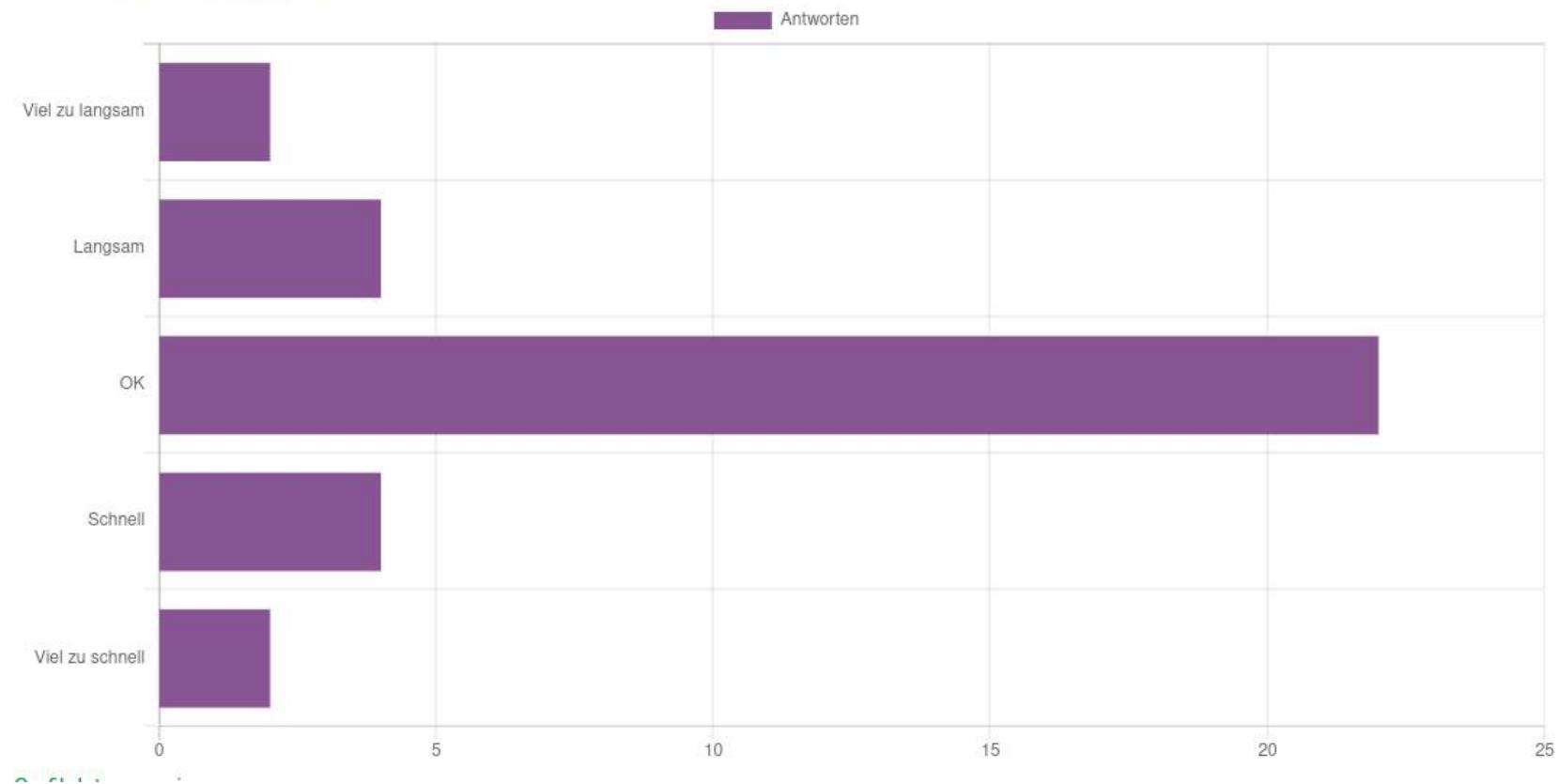
Was tut der folgende Code?

```
#include <iostream>
int main()
{
    int k = 1, nFac = 1;
    int lim = 5;
    do
    {
        nFac *= k;
        k++;
    } while (nFac < 20);
    std::cout << nFac << "\n";
}
```

Welche Zahl wird ausgegeben?

# Stand der Umfrage am 4. Nov...

## 1. Das Tempo der Vorlesung ist...



## Lektion 5

# Container std::vector, std::map, u.v.m. aus der Standard Library

```
#include <vector>
int main()
{
    std::vector<int> allSquares;
    for (int k = 0; k < 10; k++)
    {
        // add value as last element in std::vector<int> allSquares
        allSquares.push_back(k * k);
    }

    for (size_t k = 0; k < allSquares.size(); k++)
    {
        std::cout << allSquares[k] << std::endl;
    }
}
```

- codes von git herunterladen
- std::vector
- std::map
- for-Loops über Container
- Typdefinition per “auto”
- Beispiele: harmonicOscillator.cpp

# std::vector – Werte gleichen Typs in einem Objekt

```
#include <vector>
int main()
{
    std::vector<int> allSquares;
    std::cout << "initial size = " << allSquares.size() << std::endl;
    for (int k = 0; k < 10; k++)
    {
        // add value as last element in std::vector<int> allSquares
        allSquares.push_back(k * k);
    }
    std::cout << "size now = " << allSquares.size() << std::endl;
    // allSquare has indices 0,1,...,allSquares.size()-1
    for (size_t k = 0; k < allSquares.size(); k++)
    {
        std::cout << allSquares[k] << std::endl;
    }
}
```

include für Verwendung von std::vector

Typdeklaration als “vector” vom Typ “int” kann beliebig viele Variable vom Typ int enthalten

gibt =0 ... vector ist leer

hänge neues Element mit diesem Wert an

enthält jetzt 10 Elemente

for-Loop durch alle Indices

das k-te Element von allSquares

type size\_t – “size type” siehe folgendes Slide

# **size\_t – der “size type”**

## **Ich verwende size\_t für Vergleiche mit .size() hier die Gründe:**

Historisch sind die Funktionen size() in der Standard Library integers ohne Vorzeichen

dieser Typ ist zumeist  
    unsigned int  
allgemein kann man  
    size\_t  
verwenden

der Vergleich

```
int k=-1;  
k<allSquares.size();
```

kann eventuell schief gehen, wenn k als unsigned int interpretiert wird  
k hat ein sign-bit, was bei unsigned int als, z.B.  $2^{32}$  interpretiert würde

Compiler und IDEs können davor warnen

Um nicht durch unnötige Warnungen von den wichtigen abgelenkt zu werden,  
sollte man sich angewöhnen, den Typ immer geeignet zu verwenden.

# Loop über alle Elemente eines std::vector

Häufig will man den Loop über den ganzen std::vector

```
for (size_t k = 0; k < allSquares.size(); k++)
{
    std::cout << allSquares[k] << std::endl;
}
```

Für den Zweck einfachere Variante (vgl. Python):

```
std::cout << "using the for-loop over all elements of vec: ";
for (int v : allSquares)
{
    std::cout << " " << v;
}
std::cout << std::endl;
```

# auto – automatische Typdeklaration (inferred type)

Wenn der Compiler den Typ erschliessen (“infer the type”) kann...

```
std::vector<int> allSquares;
```

=> jedes Element ist vom Typ int

```
for (int v : allSquares)
```

=> v ist vom Typ int

...dann alternativ folgende Form der Typdeklaration

```
for (auto v : allSquares)
{
    std::cout << " " << v;
}
```

...oder auch so:

inferredType wird automatisch als type int deklariert

```
auto inferredType=allSquares[0];
```

# Wertzuweisung an einen std::vector

(siehe Container/detailsOnVector.cpp)

```
int main()
{
    // vector of 5, all values =3.14159
    std::vector<double> vecPi(5,3.15159);

    // vector of 3 w/o initializing (content may be random or illegal)
    //NOTE: some compilers to initialize to some value, but there is not guarantee
    //      NEVER rely on that (current g++ seems to be putting 0)
    std::vector<double> vecUnini(3);
    std::cout<<"uninitialized value: "<<vecUnini[2]<<std::endl;
    for(size_t k=0;k<vecUnini.size();k++)
    {
        vecUnini[k]=k*k;
    }

    // directly set values for vector
    std::vector<int> vecValues({1,3,5,7,11,13,17});
}
```

# !!! ACHTUNG: std::vector überprüft Index NICHT !!!

(siehe Container/detailsOnVector.cpp)

```
// this will likely run in many cases but print accidental values
for(size_t k=0;k<vecValues.size()+2;k++){
    std::cout<<" "<<vecValues[k];
}
std::cout<<std::endl;

// this almost certainly crashes the code with a "segmentation fault"
std::cout<<vecValues[INT_MAX];
```

Hier passieren sehr leicht Fehler!

## Elementzugriff mit Überprüfung: at(...)-Funktion

```
std::vector<int> vecValues({1,3,5,7,11,13,17});
// at(...) - check index value
// is will abort the code after printing 17
for(size_t k=0;k<vecValues.size()+2;k++){
    std::cout<<" "<<vecValues.at(k)<<std::endl;
}
std::cout<<std::endl;
```

Überprüfung kostet Rechenzeit (nicht zu unterschätzen)

Empfehlung: [k]-Form verwenden, Sorgfalt bei Werten von k

# std::map – beliebige Objekte als Index

```
int main()
{
    std::map<int, int> mapSquareToRoot;           type des Wertes ("value"), hier: ebenfalls int
    std::cout << "initial size = " << mapSquareToRoot.size() << std::endl;
    for (int k = 0; k < 3; k++)
    {
        // insert new element with index k*k and value k
        mapSquareToRoot[k * k] = k;
    }

    for (auto p : mapSquareToRoot)                 key      value
    {
        std::cout << p.first << " maps to integer root " << p.second << std::endl;
    }
    std::cout << "size: " << mapSquareToRoot.size() << std::endl;
}
```

Falls key noch nicht existiert, wird neues Element erzeugt:

```
std::cout << "access value undefined: " << mapSquareToRoot[7] << std::endl;
std::cout << "size has increased: " << mapSquareToRoot.size() << std::endl;
```

# ...Beispiel für ungewöhnlichen key in std::map

...verwende std::vector<double> als Index (= "key")

key: std::vector<double>

value: std::string

```
std::map<std::vector<double>, std::string> sillyMap;
```

```
// create 3 standard vectors with sizes 1,2,9, respectively
std::vector<double> v1(1,1.1),v2(2,17.),v9(9,199.);
```

```
sillyMap[v1]="value of 1.1,";
sillyMap[v2]="values, 17. each,";
sillyMap[v9]="values, 199. each";
```

```
for(auto v: sillyMap){
    std::cout<<v.first.size()<<" "<<v.second<<std::endl;
}
```

std::vector und std::map sind Beispiele für “Standard Library Containers”

Weitere Beispiele:

- std::set
- std::list
- std::deque

Container teilen viele Eigenschaften,  
z.B. folgenden Funktionen haben viele der Container:

- size() ...Anzahl der Elemente im Container
- front() ...das erste Element des Containers
- back() ...das letzte Element
- push\_back(val) ... ein Element mit Wert val anhängen
- pop\_back()... Container durch Entfernen des letzten Elements verkleinern
- clear() ...alle Elemente des Containers entfernen, danach size()==0

...usw...

Container sind sehr effizient – verwenden, wenn sie passen!

# Pausenaufgabe Nov 05 (10 min)

Der folgende Code ist fehlerhaft und braucht Ergänzung  
Notieren Sie, was nötig wäre, um guten Code zu erhalten

```
#include <vector>
int main (){
    std::vector<int> allRes;
    std::cout<<"Enter N: "
    // check input
    for(int k=0;k++){
        res+=k;
        allRes.push_back(res);
    }
    std::cout<<"what is this result?: "<<res<<std::endl;
    // use different variants of the loop
    std::cout<<"now show all results";
    for(){
    }
}
```

Von der UVB auch per git zu erhalten als pauseNov05.cpp:

```
student@EinProg:~$ git clone https://gitlab.physik.uni-muenchen.de/AG-Scrinzi/einprog.git scratch
Cloning into 'scratch'...
remote: Enumerating objects: 65, done.
remote: Counting objects: 100% (65/65), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 65 (delta 15), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (65/65), 14.23 KiB | 7.12 MiB/s, done.
Resolving deltas: 100% (15/15), done.
student@EinProg:~$ cd scratch/
student@EinProg:~/scratch$ cd Pause/
student@EinProg:~/scratch/Pause$ ls pauseNov05.cpp
pauseNov05.cpp
```

# Der harmonische Oszillator in der klassischen Mechanik

Hamiltonische Formulierung der klassischen Mechanik (in 3d)

$$\frac{d}{dt} \begin{pmatrix} \vec{x} \\ \vec{p} \end{pmatrix} = \begin{pmatrix} +\vec{\nabla}_p \\ -\vec{\nabla}_x \end{pmatrix} H(\vec{x}, \vec{p})$$

Gewöhnliche Differenzialgleichung (ordinary differential equation - ODE)

$$\dot{Y} = f(Y, t), \quad Y \dots \text{Vektor der Dimension } 3 \times 2$$

$f(Y, t) \dots \text{Funktion des Vektors } Y \text{ und der Zeit } t$

## Lösung mittels Euler-Verfahren

viele kleine Zeitschritte der Grösse  $h$

$$Y(t+h) = Y(t) + h f(Y, t)$$

beginnend bei Zeit  $t_0$  mit Anfangswert  $Y(t_0)$

bis zur Endzeit  $t_1$  mit Resultat  $\approx Y(t_1)$

# Implementierung in C++

mit Blick auf Verallgemeinerung zu höheren Dimensionen und anderen H

`std::vector<std::vector<double>> Y Vektoren x, p mit je Länge der Dimension`

`std::vector<std::vector<double>> derivative(std::vector<std::vector<double>> Y);`

Schrittgrösse h

# Der harmonische Oszillator in der klassischen Mechanik

Harmonischer Oszillator in 1d, Masse m, Hook-Konstante k

$$H(x, p) = \frac{p^2}{2m} + \frac{k}{2}x^2$$

$$\frac{d}{dt} \begin{pmatrix} x \\ p \end{pmatrix} = \begin{pmatrix} +\partial_p \\ -\partial_x \end{pmatrix} H(x, p) = \begin{pmatrix} +p/m \\ -kx \end{pmatrix}$$

**Exakte Lösung für  $p(0)=0$**

$$x(t) = x(0)\cos(\omega xt)$$

**Exakte Lösungen – sehr wichtig für Code Tests**

# Verwendung des Code-repository “git”

git (Englisch ~ “Depp”) – ein ziemlich schlaues Version-Control System  
von Linus Torvald – Entwickler von Linux

Webhosts: github, gitlab, etc.

3 Ebenen:

## **Server-repository (server repo)**

Enthält alle aktuellen Files und alle früheren Versionen  
in unserm Fall auf

<https://gitlab.physik.uni-muenchen.de/AG-Schinzi/einprog.git>  
kann nur durch Autorisierte verändert werden

## **Lokales repository (local repo)**

Liegt auf Ihrem Computer (kein Internet nötig)

Beginnt mit der vollen Info vom Server-Repo: “clone” des server-repo  
alle Files und früheren Versionen  
kann lokal verändert werden

## **Lokale Files**

Ein “normaler” Directory-Baum

Änderungen können dann mittels “commit” ins local repo integriert werden

# git clone – local repo als Kopie des server-repo erzeugen

```
student@EinProg:~$ git clone https://gitlab.physik.uni-muenchen.de/AG-Scrinzi/einprog.git CodesAgain
Cloning into 'CodesAgain'...
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 43 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (43/43), 11.72 KiB | 5.86 MiB/s, done.
Resolving deltas: 100% (4/4), done.
student@EinProg:~$ ls
a.out  CodesAgain  Documents  helloworld      helloworld.o  MyWorkDirectory  snap
Codes  Desktop    Downloads  helloworld.cpp  Libraries     Screenshots
student@EinProg:~$
```

your choice of repo directory

```
student@EinProg:~$ cd CodesAgain
student@EinProg:~/CodesAgain$ ls      Files and directories controlled by local repo
Class  Container  HarmonicOscillator  IfElse  Loops  README.md
Climits  FloatingPoint  HelloStudent  IncorrectCode  Multiply
student@EinProg:~/CodesAgain$
```

# Änderungen im lokalen Directory

Editiere, z.B. File incorrectCode.cpp

```
#include <iostream>
int main()
{
    // add one (in this case useless) line and save
    cout << "Hello Student!" << std::endl;
    return 0
};
```

## git status

Veränderungen relativ zum local repo

```
student@EinProg:~/CodesAgain/IncorrectCode$ git status
```

On branch main

Your branch is up to date with 'origin/main'.

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: incorrectCode.cpp

no changes added to commit (use "git add" and/or "git commit -a")

```
student@EinProg:~/CodesAgain/IncorrectCode$
```

# git diff und git commit

## git diff Unterschiede der Files zum local repo im Detail

```
student@EinProg:~/CodesAgain/IncorrectCode$ git diff
diff --git a/IncorrectCode/incorrectCode.cpp b/IncorrectCode/incorrectCode.cpp
index 21149ef..15a2bc5 100644
--- a/IncorrectCode/incorrectCode.cpp
+++ b/IncorrectCode/incorrectCode.cpp
@@ -6,6 +6,7 @@
 #include <iostream>
 int main()
 {
+    // add one (in this case useless) line and save
    cout << "Hello Student!" << std::endl;
    return 0
};
student@EinProg:~/CodesAgain/IncorrectCode$
```

## git commit Files in Directory ins local repo integrieren

```
student@EinProg:~/CodesAgain/IncorrectCode$ git commit -a -m"This is a git demo"
[main 939d7dc] this is a git demo
 1 file changed, 1 insertion(+)
student@EinProg:~/CodesAgain/IncorrectCode$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
student@EinProg:~/CodesAgain/IncorrectCode$
```

# git log – Geschichte des Repo anzeigen

```
commit 939d7dc3283de568d0de2d2727e48c256dd5094f (HEAD -> main)
Author: Armin Scrinzi <armin.scrinzi@lmu.de>
Date:   Tue Nov 5 08:58:02 2024 +0100
        on local repo

    this is a git demo

commit 816c5eb44996bfe6ddf32db0ef9901a7dced0ce8 (origin/main, origin/HEAD)
Author: Armin Scrinzi <armin.scrinzi@lmu.de>
Date:   Mon Nov 4 21:22:55 2024 +0100
        as pulled from server

    added HarmonicOscillator/

commit bd47e19c70cec8790d3ec07e3fe8db3bf23032cf
Author: Armin Scrinzi <armin.scrinzi@lmu.de>
Date:   Mon Nov 4 21:12:36 2024 +0100

    a few new codes

commit f32bc600f05d5061eaa0b1c8a525e9a64f10330a
Author: Armin Scrinzi <armin.scrinzi@lmu.de>
Date:   Thu Oct 24 20:02:47 2024 +0200

    adding first set of codes
```

# git pull – local repo mit server-repo vereinen

## 1. keine Änderung am Server

```
student@EinProg:~/CodesAgain/IncorrectCode$ git pull  
Already up to date.
```

## 2. erstmaliges git pull – Strategie zu Vereinigung mit server-repo wählen

```
student@EinProg:~/CodesAgain/IncorrectCode$ git pull  
remote: Enumerating objects: 5, done.  
remote: Counting objects: 100% (5/5), done.  
remote: Compressing objects: 100% (3/3), done.  
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)  
Unpacking objects: 100% (3/3), 335 bytes | 335.00 KiB/s, done.  
From https://gitlab.physik.uni-muenchen.de/AG-Scrinzi/einprog  
 816c5eb..54ee8e0 main      -> origin/main  
hint: You have divergent branches and need to specify how to reconcile them.  
hint: You can do so by running one of the following commands sometime before  
hint: your next pull:  
hint:  
hint:   git config pull.rebase false # merge (the default strategy)  
hint:   git config pull.rebase true  # rebase  
hint:   git config pull.ff only    # fast-forward only  
hint:  
hint: You can replace "git config" with "git config --global" to set a default  
hint: preference for all repositories. You can also pass --rebase, --no-rebase,  
hint: or --ff-only on the command line to override the configured default per  
hint: invocation.  
fatal: Need to specify how to reconcile divergent branches.  
student@EinProg:~/CodesAgain/IncorrectCode$
```

# git pull – aktualisieren des local repo mit Server-Repo

3. Änderung am Server im File README.md - "merge"

```
student@EinProg:~/CodesAgain/IncorrectCode$ git pull
Merge made by the 'ort' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
student@EinProg:~/CodesAgain/IncorrectCode$
```

Lokales Repo enthält jetzt die Änderung vom Server-Repo

```
student@EinProg:~/CodesAgain/IncorrectCode$ more ../README.md
## Disclaimer

Dies sind Arbeitsmaterialien zur Vorlesung "Einführung ins Pro
ind sie ohne Bedeutung. Codes sind nicht notwendigerweise kons

Dies ist ein kleiner Zusatz, um git pull zu erläutern
student@EinProg:~/CodesAgain/IncorrectCode$
```

# conflict – git pull nach Änderung am Server-Repo

```
student@EinProg:~/CodesAgain/IncorrectCode$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 470 bytes | 470.00 KiB/s, done.
From https://gitlab.physik.uni-muenchen.de/AG-Scrinzi/einprog
  54ee8e0..81e2c06  main      -> origin/main
Auto-merging IncorrectCode/incorrectCode.cpp
CONFLICT (content): Merge conflict in IncorrectCode/incorrectCode.cpp
Automatic merge failed; fix conflicts and then commit the result.
student@EinProg:~/CodesAgain/IncorrectCode$
```

## git diff – Details des conflict in incorrectCode.cpp

```
student@EinProg:~/CodesAgain/IncorrectCode$ git diff
diff --cc IncorrectCode/incorrectCode.cpp
index 15a2bc5,d47e067..0000000
--- a/IncorrectCode/incorrectCode.cpp
+++ b/IncorrectCode/incorrectCode.cpp
@@@ -6,7 -6,9 +6,13 @@
 #include <iostream>
 int main()
 {
++<<<<< HEAD
+ // add one (in this case useless) line and save
+=====
+ // here I write different line on the server in the same place
+ // as I have made changes to the local repo - this will cause a
+ // conflict when merging
++>>>>> 81e2c0639bdb256e784314d6683d1fdd720bb708
    cout << "Hello Student!" << std::endl;
    return 0
};

student@EinProg:~/CodesAgain/IncorrectCode$
```

# git status nach conflict

```
student@EinProg:~/CodesAgain/IncorrectCode$ git status
On branch main
Your branch and 'origin/main' have diverged,
and have 2 and 1 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:  incorrectCode.cpp

no changes added to commit (use "git add" and/or "git commit -a")
student@EinProg:~/CodesAgain/IncorrectCode$
```

In Editor Korrekturen vornehmen  
– insbesondere >>>>, <<<<<, ===== entfernen

```
#include <iostream>
int main()
{
    //local edit -- admit both versions:
    // add one (in this case useless) line and save
    // here I write different line on the server in the same place
    // as I have made changes to the local repo - this will cause a
    // conflict when merging
    cout << "Hello Student!" << std::endl;
    return 0
};
```

# git – nach Behebung eines conflict

Nach Editieren des Files und Behebung des Konflikts  
→ **git commit**

```
student@EinProg:~/CodesAgain/IncorrectCode$ git commit -a -m"resolved merge conflict"
[main 30303b1] resolved merge conflict
student@EinProg:~/CodesAgain/IncorrectCode$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
student@EinProg:~/CodesAgain/IncorrectCode$
```

```
commit 30303b1f8051d95c0d89cbbe6a4777f39779a973 (HEAD -> main)
Merge: 562bc5b 81e2c06
Author: Armin Scrinzi <armin.scrinzi@lmu.de>
Date: Tue Nov 5 09:26:00 2024 +0100

  resolved merge conflict

commit 81e2c0639bdb256e784314d6683d1fdd720bb708 (origin/main, origin/HEAD)
Author: Armin.Scrinzi <armin.scrinzi@physik.uni-muenchen.de>
Date: Tue Nov 5 08:18:53 2024 +0000

  Update incorrectCode.cpp

commit 562bc5bbfc2a5bde8bdc552d9e82b2f6dca9b921
Merge: 939d7dc 54ee8e0
Author: Armin Scrinzi <armin.scrinzi@lmu.de>
Date: Tue Nov 5 09:12:28 2024 +0100

  Merge branch 'main' of https://gitlab.physik.uni-muenchen.de/AG-Scrinzi/einprog

commit 54ee8e0725a5ab64da1da43becd3c1066fde4639
Author: Armin.Scrinzi <armin.scrinzi@physik.uni-muenchen.de>
Date: Tue Nov 5 08:08:03 2024 +0000
```

**...weitere Fragen?**

→ **Tutorien**

→ **Web**